Python IP Class Notebook

Nebhrajani A. V.

Contents

1 Preamble

1.1 Notebook Conventions

All code in this notebook is in Python unless specified otherwise. All code is syntax-highlighted, placed in boxes, and is line numbered. The output of the interpreter on `stdout` is printed directly below it, `verbatim`, thus.

```
1  # Print Hello world!
2  print("Hello world!")
```

```
Hello world!
```

It is recommended that you navigate using the hyperlinked TOC or the Adobe Bookmarks tree.

1.2 Hardware and Software Used

This notebook is written in an `org-mode` file and exported via LaTeX, Org version 9.3.6 on GNU Emacs 25.2.2 (x86_ 64-pc-linux-gnu, GTK+ Version 3.22.21) of 2017-09-23, modified by Debian, on a Foxconn Core i7 NanoPC running Linux Mint 19.3 XFCE 64-bit. Python 2.7.17 of 2020-04-15 is used throughout unless specified otherwise. For the Org or LaTeX source, contact aditya.v.nebhrajani@gmail.com.

1.3 Acknowledgements

I am grateful to the FSF, the GNU Project, the Linux foundation, the Emacs, StackExchange and FLOSS communities, and my father, who taught me that a world outside commercialized technology does exist and thrive.

2 NumPy

2.1 Worksheet 2020-07-26

1. Create an ndarray with values ranging from 10 to 49 each spaced with a difference of 3.

```
import numpy as np
arr=np.arange(10,50,3,dtype=int)
print(arr)
```

```
[10 13 16 19 22 25 28 31 34 37 40 43 46 49]
```

2. Find the output of the following Python code:

```
x="hello world"
print(x[:2],x[:-2],x[-2:])
```

```
('he', 'hello wor', 'ld')
```

3. Predict the output of the following code fragments:

```
import numpy as np
x=np.array([1,2,3])
y=np.array([3,2,1])
z=np.concatenate([x,y])
print(z)
```

```
[1 2 3 3 2 1]
```

4. Consider following two arrays: Array1= array([0,1,2],[3,4,5],[6,7,8]]) and Array2= array([10,11,12],[13,14,15],[16,17,18]]). Write NumPy command to concatenate Array1 and Array2:

   (a) Row wise

```
import numpy as np
Array1= np.array([[0,1,2],[3,4,5],[6,7,8]])
Array2= np.array([[10,11,12],[13,14,15],[16,17,18]])
rarr=np.concatenate([Array1,Array2],axis=1)
print(rarr)
```

```
[[ 0  1  2 10 11 12]
 [ 3  4  5 13 14 15]
 [ 6  7  8 16 17 18]]
```

(b) Column wise

```
import numpy as np
Array1= np.array([[0,1,2],[3,4,5],[6,7,8]])
Array2= np.array([[10,11,12],[13,14,15],[16,17,18]])
carr=np.concatenate([Array1,Array2],axis=0)
print(carr)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [10 11 12]
 [13 14 15]
 [16 17 18]]
```

5. To create sequences of numbers, NumPy provides a function (a)arange analogous to range that returns arrays instead of lists.

6. Find the output of following program.

```
import numpy as np
a=np.array([30,60,70,30,10,86,45])
print(a[-2:6])
```

```
[86]
```

7. Write a NumPy program to create a 2d array with 1 on the border and 0 inside.

```
import numpy as np
x = np.ones((5,5))
print("Original array:")
print(x)
print("1 on the border and 0 inside in the array")
x[1:-1,1:-1] = 0
print(x)
```

```
Original array:
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
```

```
 [1. 1. 1. 1. 1.]]
1 on the border and 0 inside in the array
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
```

8. Given following ndarray A: ([[2, 4, 6], [7, 8, 9], [1, 2, 3]]) Write the python statements to perform the array slices in the way so as to extract first row and second column.

```
import numpy as np
A = np.array([[2,4,6],[7,8,9],[1,2,3]])
print(A[0,:])
print(A[:,1])
```

```
[2 4 6]
[4 8 2]
```

9. Write python statement to create a two- dimensional array of 4 rows and 3 columns. The array should be filled with ones.

```
import numpy as np
x = np.ones((4,3))
print(x)
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

10. Find the output of following program.

```
import numpy as np
d = np.array([10,20,30,40,50,60,70])
print(d[-5:])
```

```
[30 40 50 60 70]
```

11. State at least two differences between a NumPy array and a list

NumPy Array	List
By default, numpy arrays are homogeneous	They can have elements of different data types
Element-wise operations are possible	Element-wise operations don't work on lists
They take up less space	They take up more space

12. Find the output of following program.

```
import numpy as np
d=np.array([10,20,30,40,50,60,70])
print(d[-1:-4:-1])
```

```
[70 60 50]
```

13. Write the output of the following code.

```
import numpy as np
a=[[1,2,3,4],[5,6,7,8]]
b=[[1,2,3,4],[5,6,7,8]]
n=np.concatenate((a, b), axis=0)
print(n[1])
print(n[1][1])
```

```
[5 6 7 8]
6
```

14. Which of the following is contained in NumPy library?

    (a) **N-Dimensional Array Object**
    (b) Series
    (c) DataFrame
    (d) Plot

15. Point out the correct statement:

    (a) NumPy main object is the homogeneous multidimensional array
    (b) In Numpy, dimensions are called axes
    (c) NumPy array class is called ndarray
    (d) **All of the above**

16. When the fromiter() is preferred over array()? **A:** Fromiter() is preferred over array()for creating non-numeric sequences like strings and dictionaries.

17. What is the purpose of order argument in empty(). What do 'C' and 'F' stands for? What is the default value of order argument? **A:** The "order" argument arranges the elements of the array row-wise or column-wise. C order arranges elements column wise and means "c"-like, whereas F order arranges elements row wise and means "fortran"-like. Default value of order argument is C.

18. Differentiate split() from hsplit() and vsplit(). **A:** Split() function is a general function which can be used to split an array in numpy both horizontally and vertically by providing an axis. If the axis is 0 it is the same as hsplit() and if the axis is 1 it behaves as vsplit(). The difference between split() and hsplit(),vsplit() is that split() allows you to specify the axis that you wish, and hsplit() and vsplit() are for specific axes.

19. Find the output:

(a)
```
import numpy as np
a = np.linspace(2.5,5,6)
print(a)
```

```
[2.5 3.  3.5 4.  4.5 5. ]
```

(b)
```
import numpy as np
a=np.array([[0,2,4,6],[8,10,12,14],[16,18,20,22],[24,26,28,30]])
print(a)
print(a[:3,3:])
print(a[1::2,:3])
print(a[-3:-1,-4::2])
print(a[::-1,::-1])
```

```
[[ 0  2  4  6]
 [ 8 10 12 14]
 [16 18 20 22]
 [24 26 28 30]]
[[ 6]
 [14]
 [22]]
[[ 8 10 12]
 [24 26 28]]
[[ 8 12]
 [16 20]]
[[30 28 26 24]
 [22 20 18 16]
 [14 12 10  8]
 [ 6  4  2  0]]
```

3 Pandas

3.1 Series

```
# Import numpy and pandas
import pandas as pd
import numpy as np
```

```
4
5   # Create an empty series
6   s = pd.Series()
7   print(s)
8
9   # Series from ndarray
10  data = np.array(['a', 'b', 'c', 'd'])
11
12  ## Without index
13  s = pd.Series(data)
14  print(s)
15  ## With index
16  s = pd.Series(data, index = [100, 101, 102, 103])
17  print(s)
18
19  # Scalar series
20  s = pd.Series(5, index = [0, 1, 2, 3])
21  print(s)
22
23  # Series from dictionary
24  data = {'a' : 0., 'b' : 1., 'c' : 2.}
25
26  ## Without index
27  s = pd.Series(data)
28  print(s)
29  ## With index
30  s = pd.Series(data, index = ['b', 'c', 'd', 'a'])
31  print(s)
32
33  # Another dictionary example
34  f_dict = {'apples': 500, 'kiwi': 20, 'oranges': 100, 'cherries': 6000}
35  print(f_dict)
36
37  arr = pd.Series(f_dict)
38  print('\nArray Items')
39  print(arr)
```

```
Series([], dtype: float64)
0    a
1    b
2    c
3    d
dtype: object
100    a
101    b
102    c
103    d
dtype: object
```

```
0    5
1    5
2    5
3    5
dtype: int64
a    0.0
b    1.0
c    2.0
dtype: float64
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
{'kiwi': 20, 'cherries': 6000, 'apples': 500, 'oranges': 100}

Array Items
apples       500
cherries    6000
kiwi          20
oranges      100
dtype: int64
```

```python
# Indexing
import pandas as pd
from pandas import Series
arr = Series([22, 44, 66, 88, 108])
print(arr[[1, 3, 0, 4]])
```

```
1     44
3     88
0     22
4    108
dtype: int64
```

```python
import pandas as pd
ds1 = pd.Series([2, 4, 6, 8, 10])
ds2 = pd.Series([1, 3, 5, 7, 9])
print(ds1)
print(ds2)
ds = ds1 + ds2
print("Add two Series:")
print(ds)
print("Subtract two Series:")
ds = ds1 - ds2
print(ds)
print("Multiply two Series:")
```

```
ds = ds1 * ds2
print(ds)
print("Divide Series1 by Series2:")
ds = ds1 / ds2
print(ds)
```

```
0     2
1     4
2     6
3     8
4    10
dtype: int64
0    1
1    3
2    5
3    7
4    9
dtype: int64
Add two Series:
0     3
1     7
2    11
3    15
4    19
dtype: int64
Subtract two Series:
0    1
1    1
2    1
3    1
4    1
dtype: int64
Multiply two Series:
0     2
1    12
2    30
3    56
4    90
dtype: int64
Divide Series1 by Series2:
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.111111
dtype: float64
```